ALPHABY

# Training report M1:

*SCHUFFENECKER Antoine*
Roll: *Intern*
Class: *Master 1 Csmi*
Session: *2020/2021*
Email: *antoine.schuffenecker@gmail.com*

Course: *PROJET* – Teacher: *Cristhophe PRUD'HOMME*
Submission date: *23/08*

## Contents

I) Introduction 1

1.Context: 1

2.Supervisors: 1

3.Subject: 1

II) Organisation 2

1.Roadmap: 2

2.Communication: 2

III) Training and first results 3

III) Presentation of the problem 9

IV) First Approach 9

1.Extraction 9

2.Issues: 13

V) Second Approach 14

1.Extraction.v2: 14

2.Results: 15

VI) Conclusion 20

VII) Acknowledgment 20

# I) Introduction

## 1.Context:

This internship is part of the first year of the CSMI master's degree. From May 31 to July , I did my internship in the company Alphaby based in Nancy,France . I did my internship in teleworking. This internship gave me an opportunity to improve my skills in data science, data analysis and some new tools that I discovered. The project was about using data from job offers from the Pole emploi api.

## 2.Supervisors:

My supervisor was Yacine ABBOUD. He has a thesis in data science and is the co-founder of Alphaby. I had another supervisor in the company Adrien ROUGERON, he helped me a lot at the start to explain which tools I would be using and helped me when I met some dead-ends during the project.

## 3.Subject:

The project I got is an internal development project which wants to use the data gathered in the job offers given by companies. Its different goals are yet to be defined and adjusted but the main ideas are to help companies to redact job offers to be more precise and complete but also more pertinent. For example in the technology field some of the recruiters might not search for the right job given their expectations. My part of the job was to analyze and find information about the data.

# II) Organisation

## 1.Roadmap:

Goals:

-learning how to use a mongoDB Database with the tool mongoCompass

-learning how to use the library pymongo to exploit the Database in a python Notebook

-expanding my usage of the library panda to exploit the data

-gathering information on the data

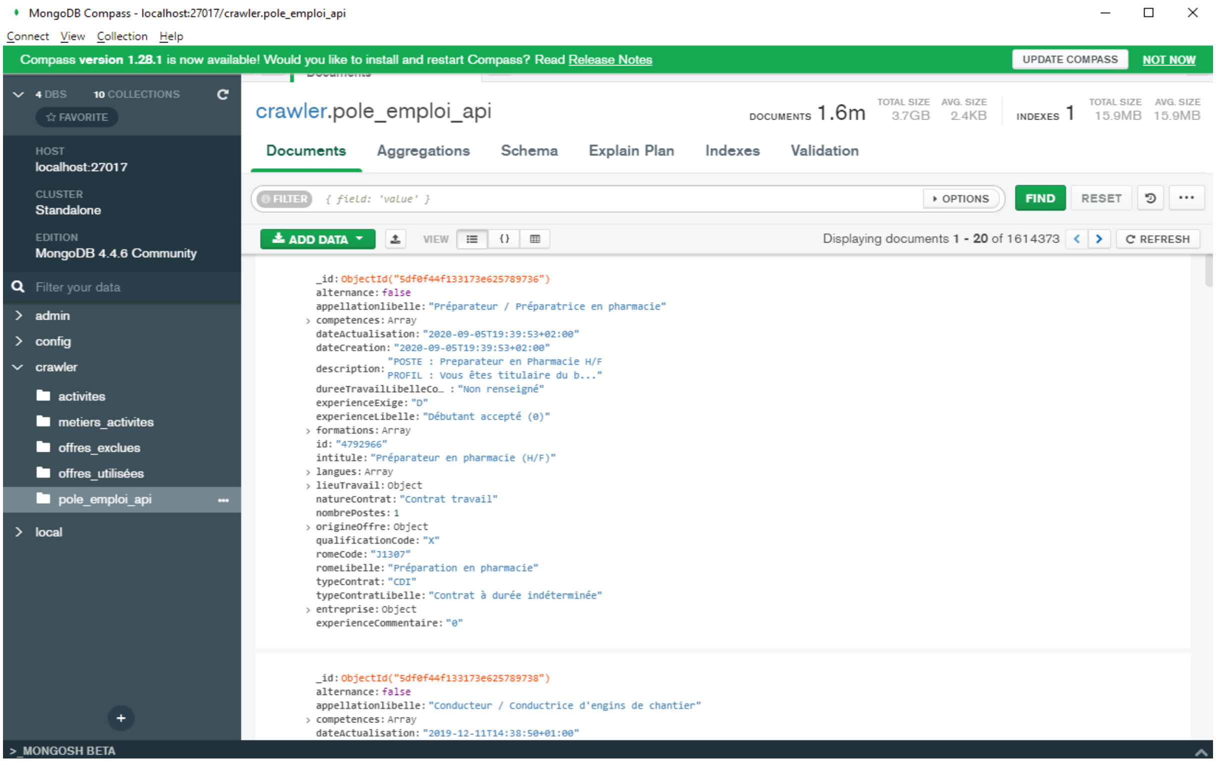
-learning how to use plotly to give my results to my supervisors

## 2.Communication:

To communicate with my supervisors I used mostly discord, for the writing it was a channel on a dedicated server , for the meetings we used teams but also a dedicated channel on discord, these meetings were usually Monday and Friday. But there was always a quick meeting 1 or 2 times the other days to explain any advance, issue or ideas. I also had a physical meeting the last week with Yacine ABBOUD to talk about my internship and the results that I got during it.

# III) Training and first results

This project used a mongoDB database, this is a nosql database which improves the usage of data which are not pre-process before the gathering. Any data gathered is stored in the form of items with different properties in a database called collection. I then used some pymongo actions to import these collections and pandas to put them into data frame to process it. The main advantage of mongoDB is that pandas is really effective to process the collection due to the way is has been developed.



```
  _id: ObjectId("5df0f44f133173e625789736")
  alternance: false
  appellationlibelle: "Préparateur / Préparatrice en pharmacie"
> competences: Array
  dateActualisation: "2020-09-05T19:39:53+02:00"
  dateCreation: "2020-09-05T19:39:53+02:00"
  description: "POSTE : Preparateur en Pharmacie H/F
               PROFIL : Vous êtes titulaire du b..."
  dureeTravailLibelleCo… : "Non renseigné"
  experienceExige: "D"
  experienceLibelle: "Débutant accepté (0)"
> formations: Array
  id: "4792966"
  intitule: "Préparateur en pharmacie (H/F)"
> langues: Array
> lieuTravail: Object
  natureContrat: "Contrat travail"
  nombrePostes: 1
> origineOffre: Object
  qualificationCode: "X"
  romeCode: "J1307"
  romeLibelle: "Préparation en pharmacie"
  typeContrat: "CDI"
  typeContratLibelle: "Contrat à durée indéterminée"
> entreprise: Object
  experienceCommentaire: "0"

  _id: ObjectId("5df0f44f133173e625789738")
  alternance: false
  appellationlibelle: "Conducteur / Conductrice d'engins de chantier"
> competences: Array
  dateActualisation: "2019-12-11T14:38:50+01:00"
```

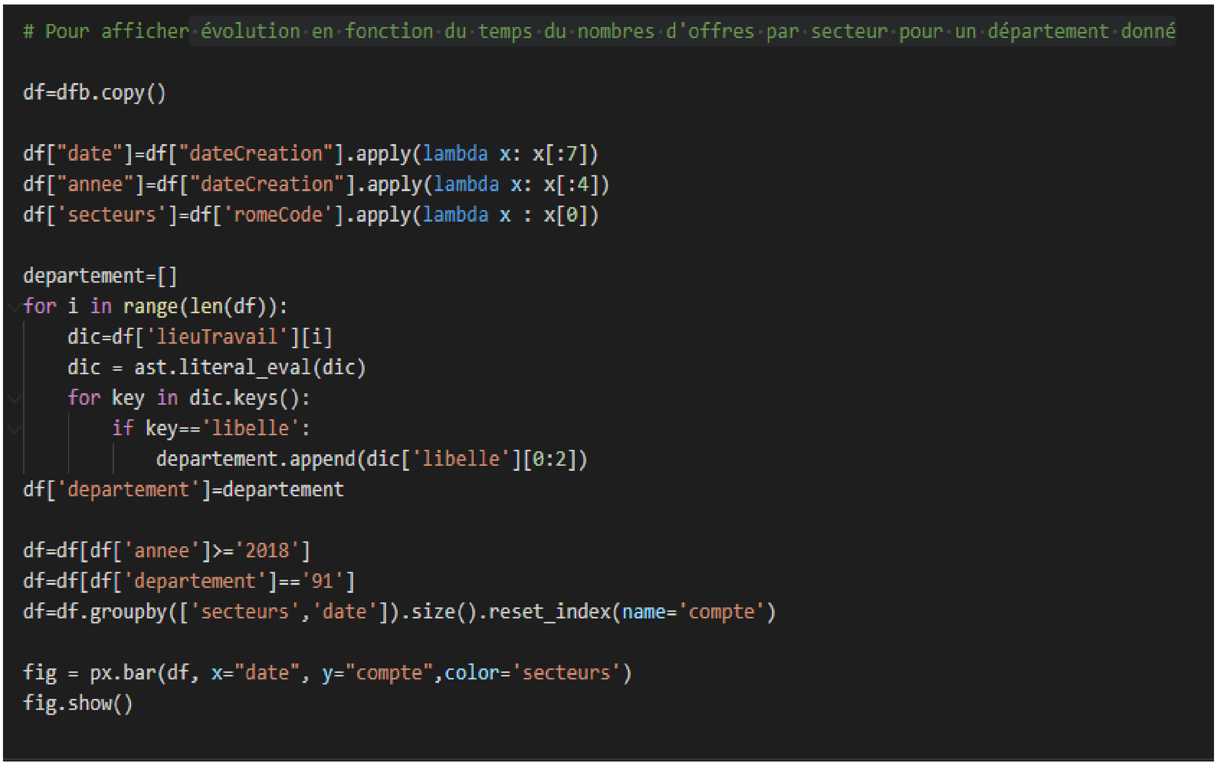
This is a mongoDB database, called collection, viewed with the tool Compass.

There are a lot of properties of these items but here at start we will use the date, the romeCode, the array "lieuTravail". And for this time only I used only a batch of 10 000 offers of the 1,6 millions that I got.

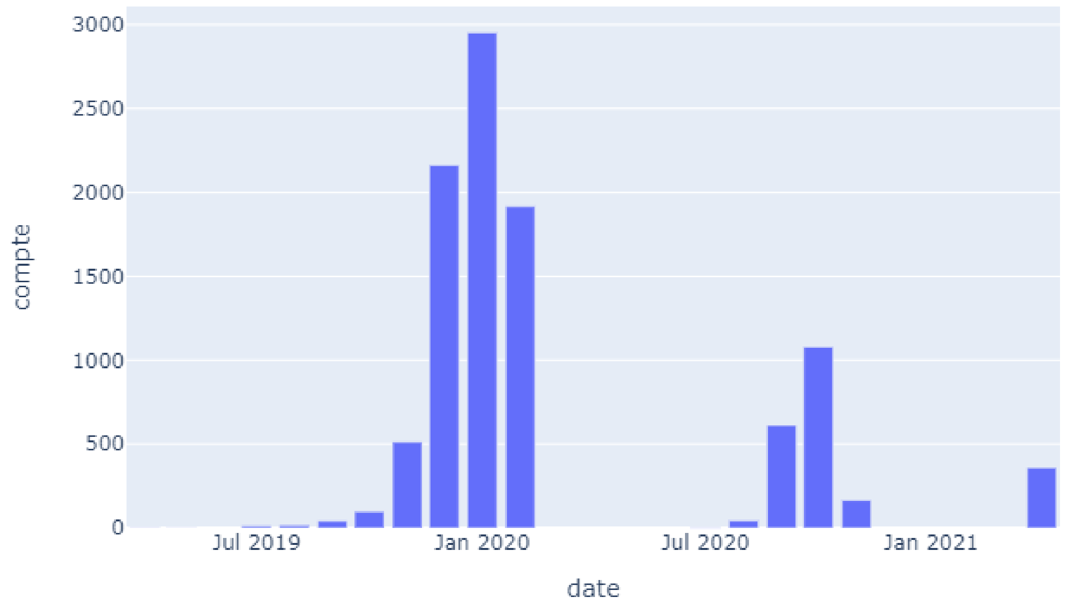
The romeCode is a code used by Pole Emploi to classify any job listed, the first letter is the sector of the job , then the two numbers specify the domain, finally the last number gives the job. For example A1201 is a A for agriculture (sector), A12 gives "Agriculture/élevage, le domaine Forêts/espaces naturels" and A1201 is giving us the job "bucheron, élagueur,commis de coupe".

These are the first results that I got. I tried to get an evolution of the offers to know if there were any months where the offers would increase or not. There is clearly a pattern but the most impressive thing was that during covid no offers were emitted. Of course it does not mean that no one did any offers, but the data were not collected during this time.

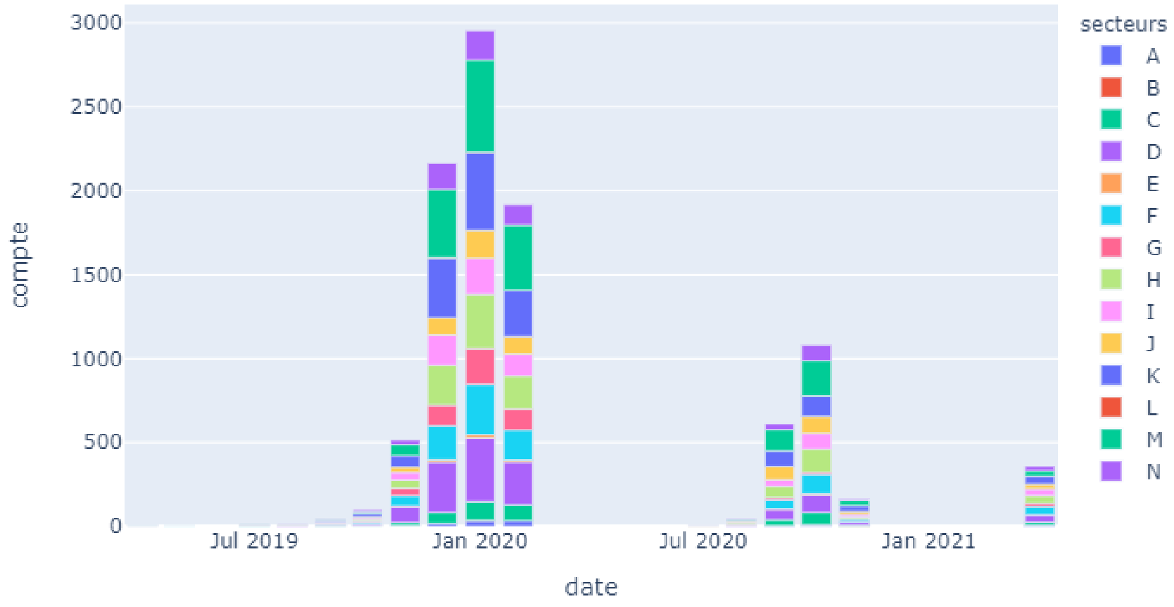
The code I created was for all results shown here, resembling the next one with slight changes to get the information that I wanted.



```
# Pour afficher évolution en fonction du temps du nombres d'offres par secteur pour un département donné

df=dfb.copy()

df["date"]=df["dateCreation"].apply(lambda x: x[:7])
df["annee"]=df["dateCreation"].apply(lambda x: x[:4])
df['secteurs']=df['romeCode'].apply(lambda x : x[0])

departement=[]
for i in range(len(df)):
    dic=df['lieuTravail'][i]
    dic = ast.literal_eval(dic)
    for key in dic.keys():
        if key=='libelle':
            departement.append(dic['libelle'][0:2])
df['departement']=departement

df=df[df['annee']>='2018']
df=df[df['departement']=='91']
df=df.groupby(['secteurs','date']).size().reset_index(name='compte')

fig = px.bar(df, x="date", y="compte",color='secteurs')
fig.show()
```

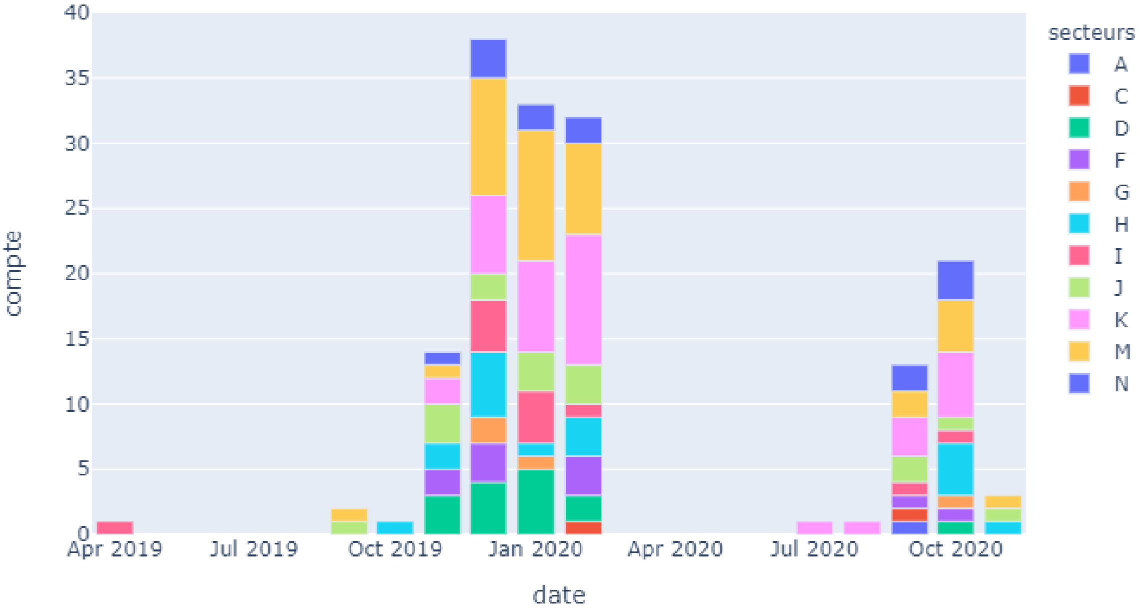


This is the evolution of the numbers of offers.

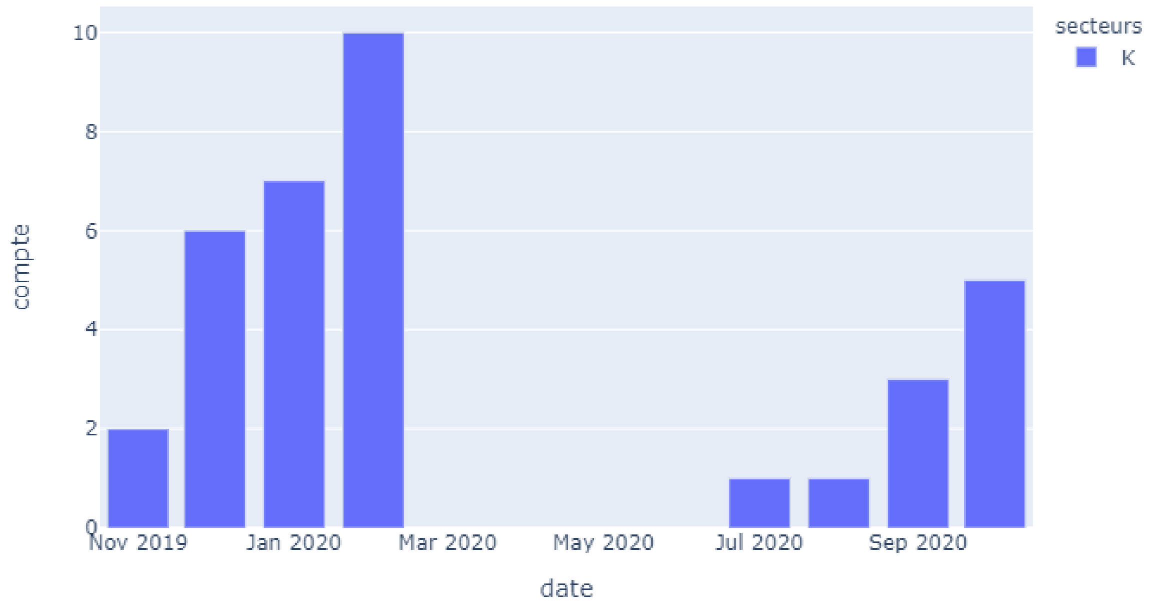


This is the evolution of the numbers of offers for each sector.

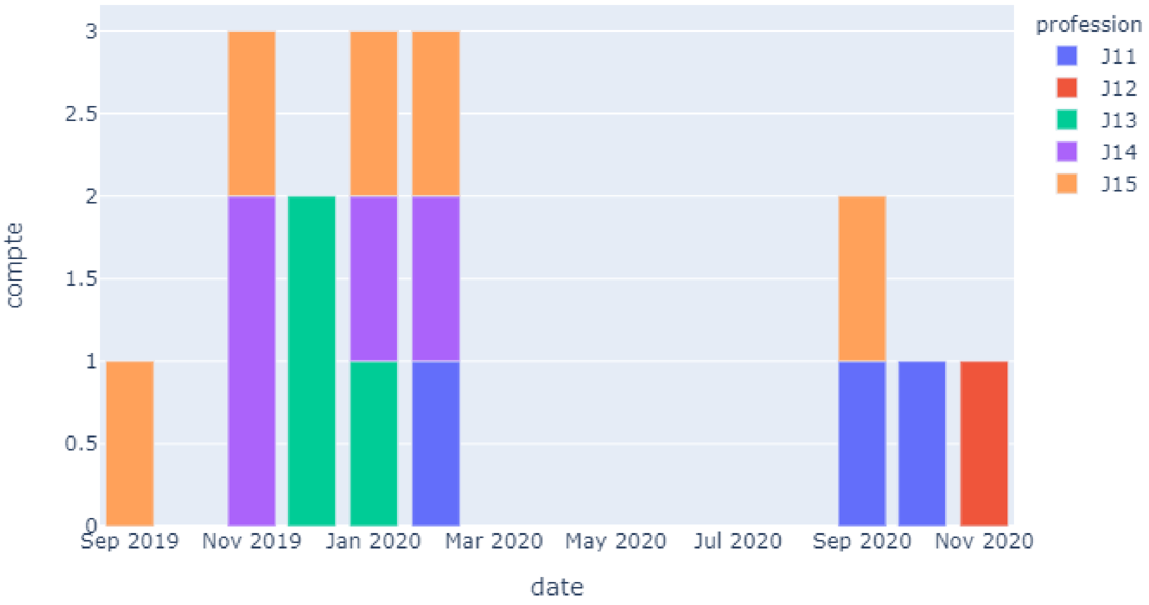
This is the same data but with one more information the sector in which the offers are targeted. To get this I retrieved the romeCode and extracted the sector. With the same principle I then manage to get data from a specific department, the different domains in a specific sector, and different professions in a sector.



This is the evolution of the numbers of offers for each sector in the department 91.



This is the evolution of the numbers of offers for the main sector in the department 91.



This is the evolution of the numbers of offers in each profession for the sector J in the department 91.

This part of the project helped me understand how the mongo database can be used, and how to extract any information, that I needed. But also to search and find the right way to show the results.

# III) Presentation of the problem

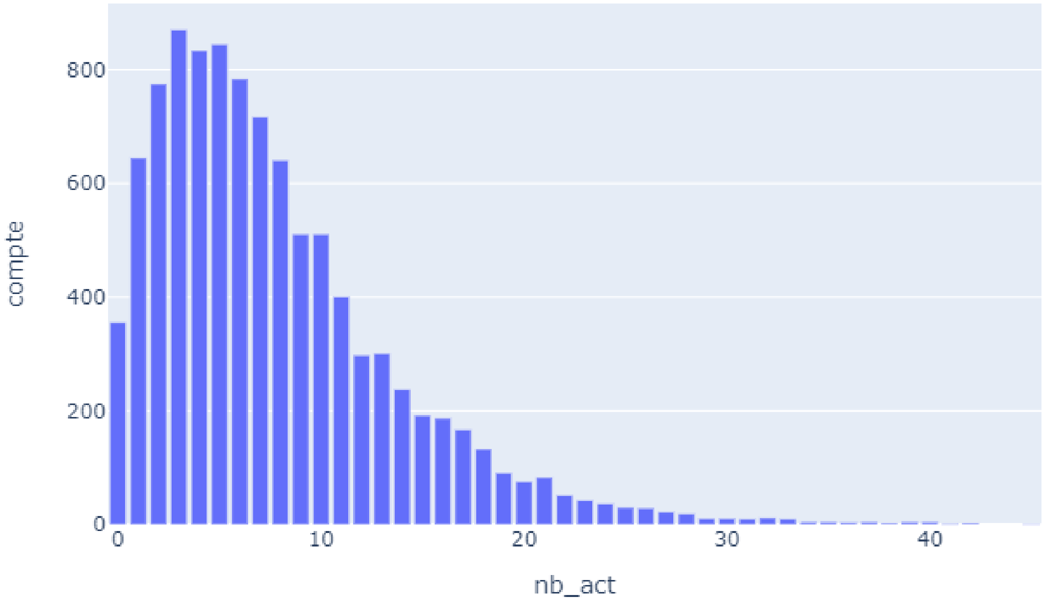
The idea of the project forced us to identify any competences which are researched by the employer and needed for the job. But more than just competences are also the different actions that the worker will perform during the job. The conclusion was to use the term “activities” to regroup all these concepts.

# IV) First Approach

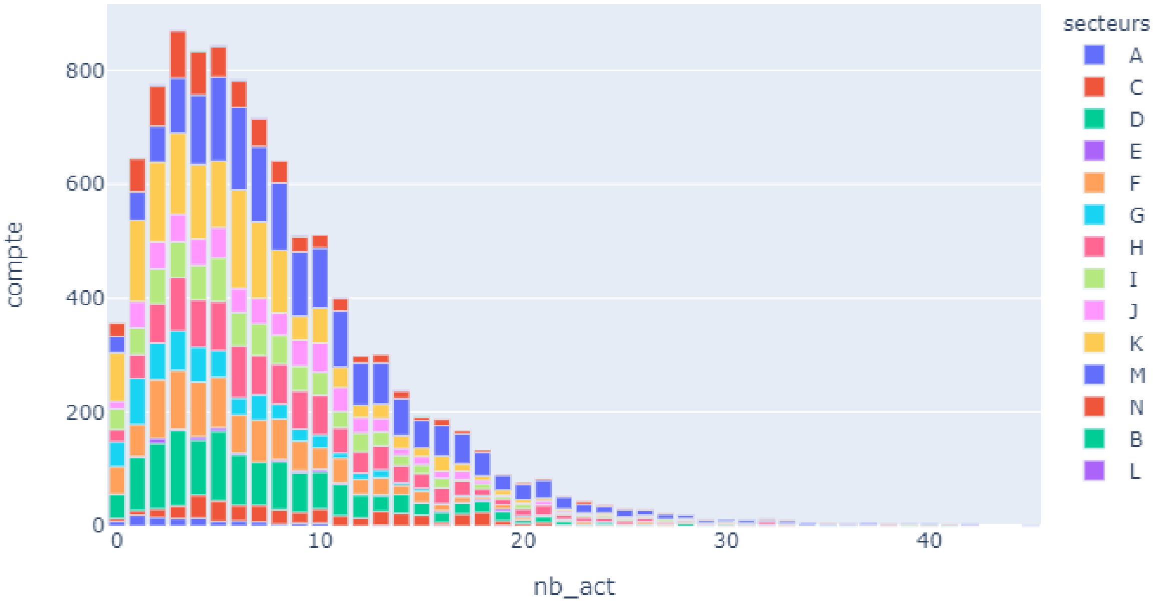
## 1.Extraction

The first idea was to gather the activities, by using any action verb which were placed in the first place of a sentence( or after a comma or a dash). For example “Trier les déchets”. Then we improved this method a bit by adding the nomination as well (here it would be “triage des déchets”). For intellectual properties issues I can not show the codes used to extract the activities. The methodology is to split the description to get all the sub sentences in it. Then the sub sentences are stored in a list. In each of these sub sentences we test if the condition ”having an action verb or nomination in first place”. For each activities found we create a new couple job/activities.

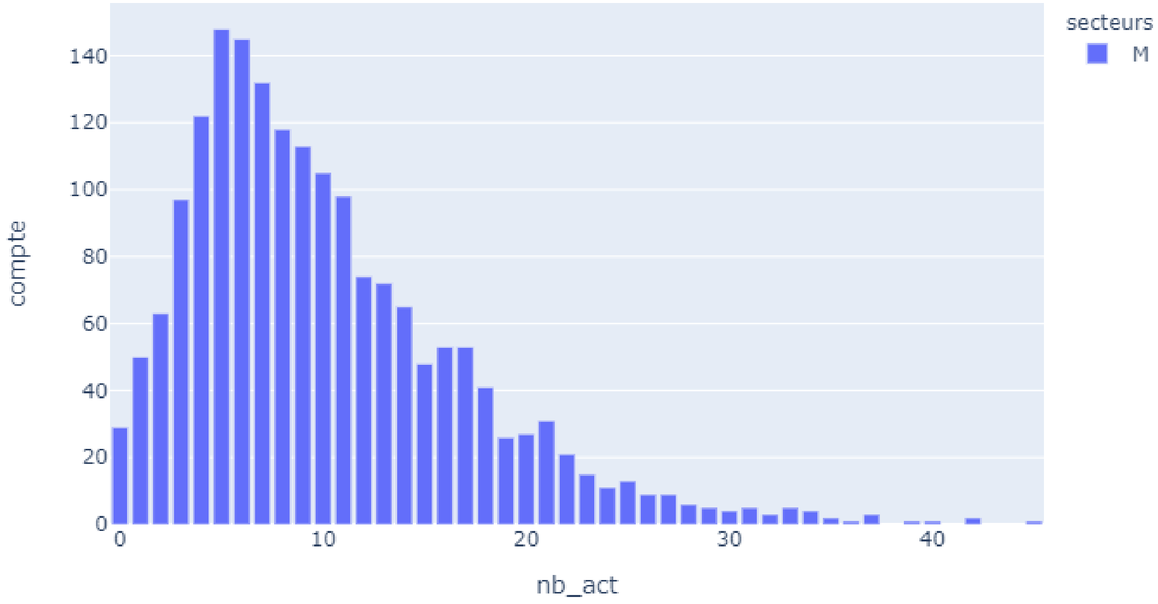
I then search several information on a batch of 10 000 offers.



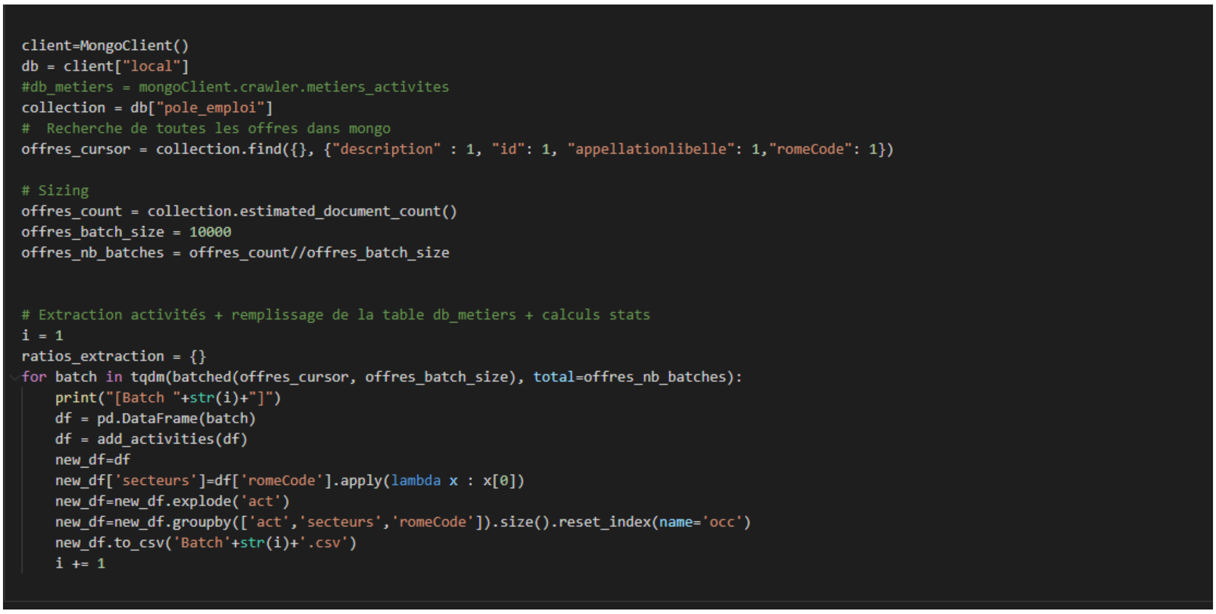
This is the histogram of the number of activities found in the offers.



This is the histogram of the number of activities found in the offers for each sector.

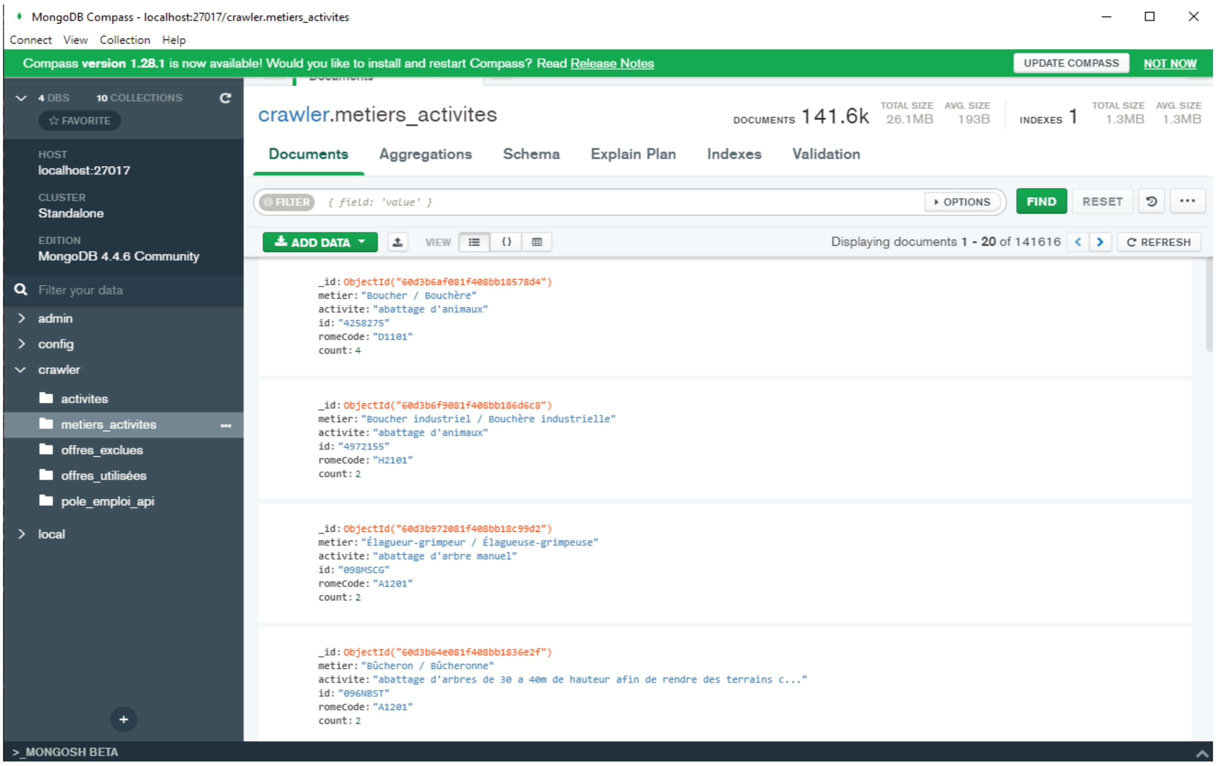


This is the histogram of the number of activities found in the offers of a specific sector.



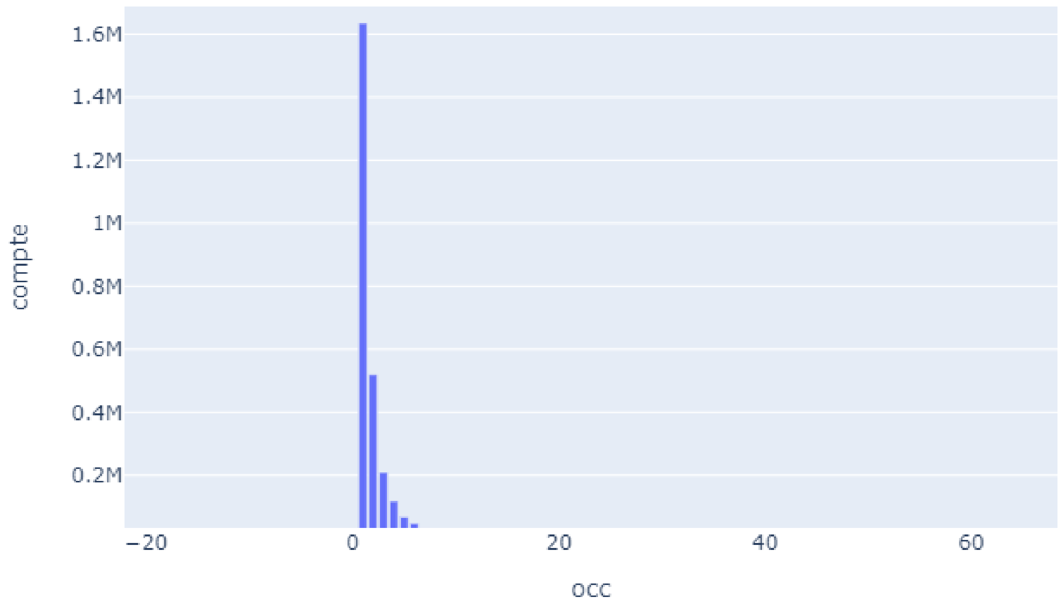
```
client=MongoClient()
db = client["local"]
#db_metiers = mongoClient.crawler.metiers_activites
collection = db["pole_emploi"]
#  Recherche de toutes les offres dans mongo
offres_cursor = collection.find({}, {"description" : 1, "id": 1, "appellationlibelle": 1,"romeCode": 1})

# Sizing
offres_count = collection.estimated_document_count()
offres_batch_size = 10000
offres_nb_batches = offres_count//offres_batch_size


# Extraction activités + remplissage de la table db_metiers + calculs stats
i = 1
ratios_extraction = {}
for batch in tqdm(batched(offres_cursor, offres_batch_size), total=offres_nb_batches):
    print("[Batch "+str(i)+"]")
    df = pd.DataFrame(batch)
    df = add_activities(df)
    new_df=df
    new_df['secteurs']=df['romeCode'].apply(lambda x : x[0])
    new_df=new_df.explode('act')
    new_df=new_df.groupby(['act','secteurs','romeCode']).size().reset_index(name='occ')
    new_df.to_csv('Batch'+str(i)+'.csv')
    i += 1
```

With this extraction I hoped, i could now process the whole data by batch, I divided the 1,6 million offers in 162 batch of 10 000 offers. And as a result I obtained a new collection (database) with all activities extracted from the offers. This collection also regroup the couple activities/job and a new attribute count to have an idea of their presence in the data.



```
_id: ObjectId("60d3b6af081f408bb18578d4")
metier: "Boucher / Bouchère"
activite: "abattage d'animaux"
id: "4258275"
romeCode: "D1101"
count: 4

_id: ObjectId("60d3b6f9081f408bb186d6c8")
metier: "Boucher industriel / Bouchère industrielle"
activite: "abattage d'animaux"
id: "4972155"
romeCode: "H2101"
count: 2

_id: ObjectId("60d3b972081f408bb18c99d2")
metier: "Élagueur-grimpeur / Élagueuse-grimpeuse"
activite: "abattage d'arbre manuel"
id: "098MSCG"
romeCode: "A1201"
count: 2

_id: ObjectId("60d3b64e081f408bb1836e2f")
metier: "Bûcheron / Bûcheronne"
activite: "abattage d'arbres de 30 a 40m de hauteur afin de rendre des terrains c..."
id: "096NBST"
romeCode: "A1201"
count: 2
```

But unfortunately it did not worked because of the issues stated later. I only manage to get a rough estimation of the distribution of the occurrence of each activities.



## 2.Issues:

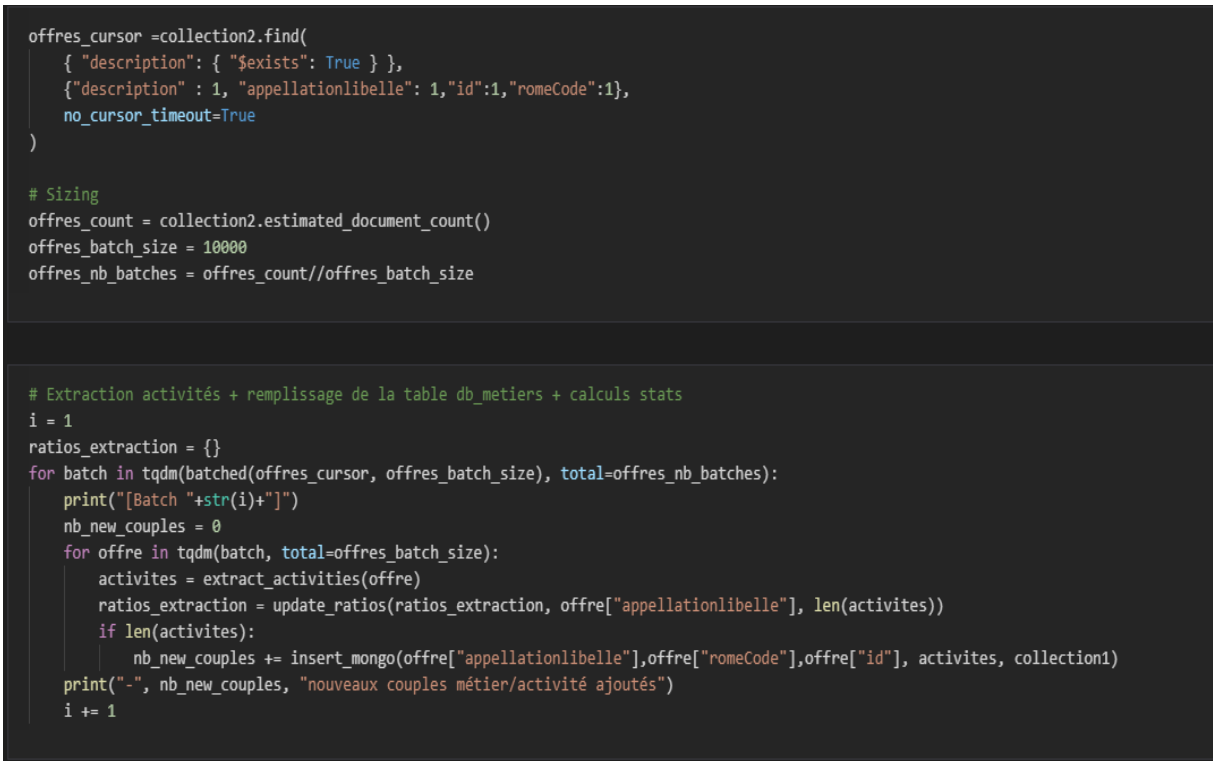
There are several issue with this method of selection of the activities. It appears that using only the start of the sentence does not give sufficient data and a huge % of the offers are not used to for more than 1 activities. Also choosing only the start works when there is an enumeration of the tasks but not when it becomes more specific. The main other issue is that the activities that are extract are not relevant since they only appear once so we need to improve the extraction

There were also issue with the way of extracting the couples and counting them. At start for any couple found, we needed to access the database that we were creating if it was found add 1 to the attribute ”count” and if not add it. So it takes approximately 50 min to process the whole 1,6 million offers. It is acceptable since in theory it is only a one time operation but with all the adjustment that are needed it is far too long.

# V) Second Approach

## 1.Extraction.v2:

This new method is way more efficient because instead of focusing on only the first word of the sentence, it split the whole description (attribute of the offers) in phrases. Then it clean the phrases of specific words or terms that were deemed not relevant in such research. Then check if the phrase is an activities with the definition of having a verb(or nomination) but not anymore on the first place. The important thing is that at the end of the operation the activities is instantly added to a collection in mongo which fasten the process a lot. Later the same activities will be grouped with a mongo operation called aggregation which is nearly instant. The new collection that I got is not very different from the previous one (in a structural way) but everything was more precise and faster so i could work way easily.

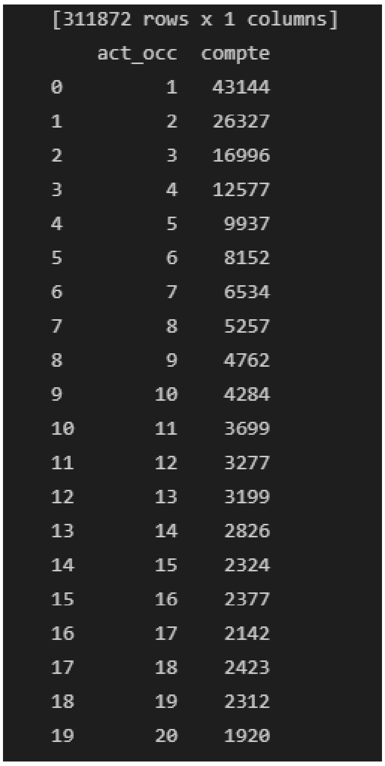


```
offres_cursor =collection2.find(
    { "description": { "$exists": True } },
    {"description" : 1, "appellationlibelle": 1,"id":1,"romeCode":1},
    no_cursor_timeout=True
)

# Sizing
offres_count = collection2.estimated_document_count()
offres_batch_size = 10000
offres_nb_batches = offres_count//offres_batch_size
```

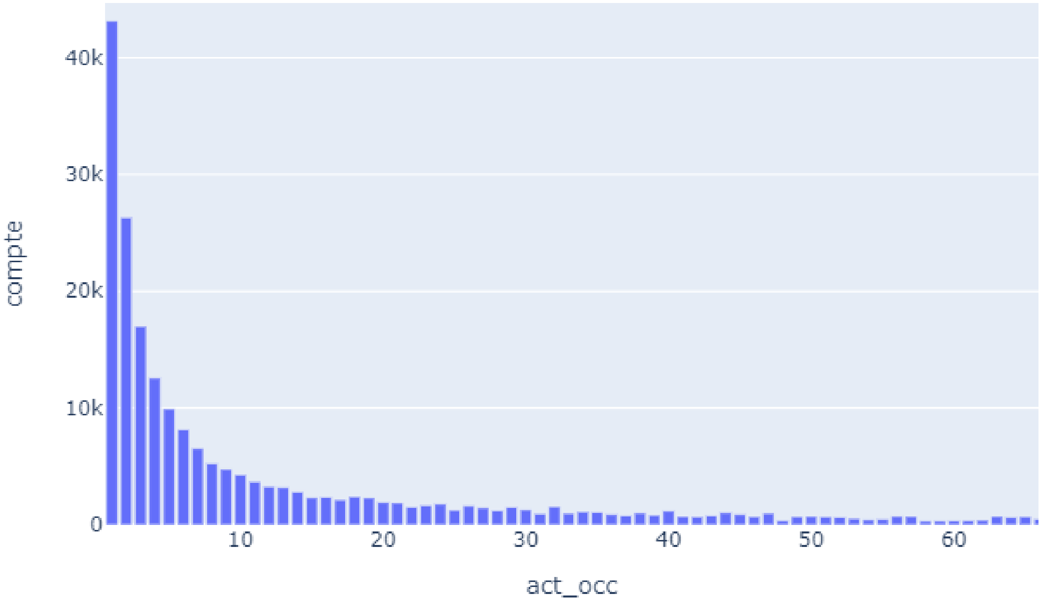
```
# Extraction activités + remplissage de la table db_metiers + calculs stats
i = 1
ratios_extraction = {}
for batch in tqdm(batched(offres_cursor, offres_batch_size), total=offres_nb_batches):
    print("[Batch "+str(i)+"]")
    nb_new_couples = 0
    for offre in tqdm(batch, total=offres_batch_size):
        activites = extract_activities(offre)
        ratios_extraction = update_ratios(ratios_extraction, offre["appellationlibelle"], len(activites))
        if len(activites):
            nb_new_couples += insert_mongo(offre["appellationlibelle"],offre["romeCode"],offre["id"], activites, collection1)
    print("-", nb_new_couples, "nouveaux couples métier/activité ajoutés")
    i += 1
```

## 2.Results:

I will only show the useful results that I obtained:

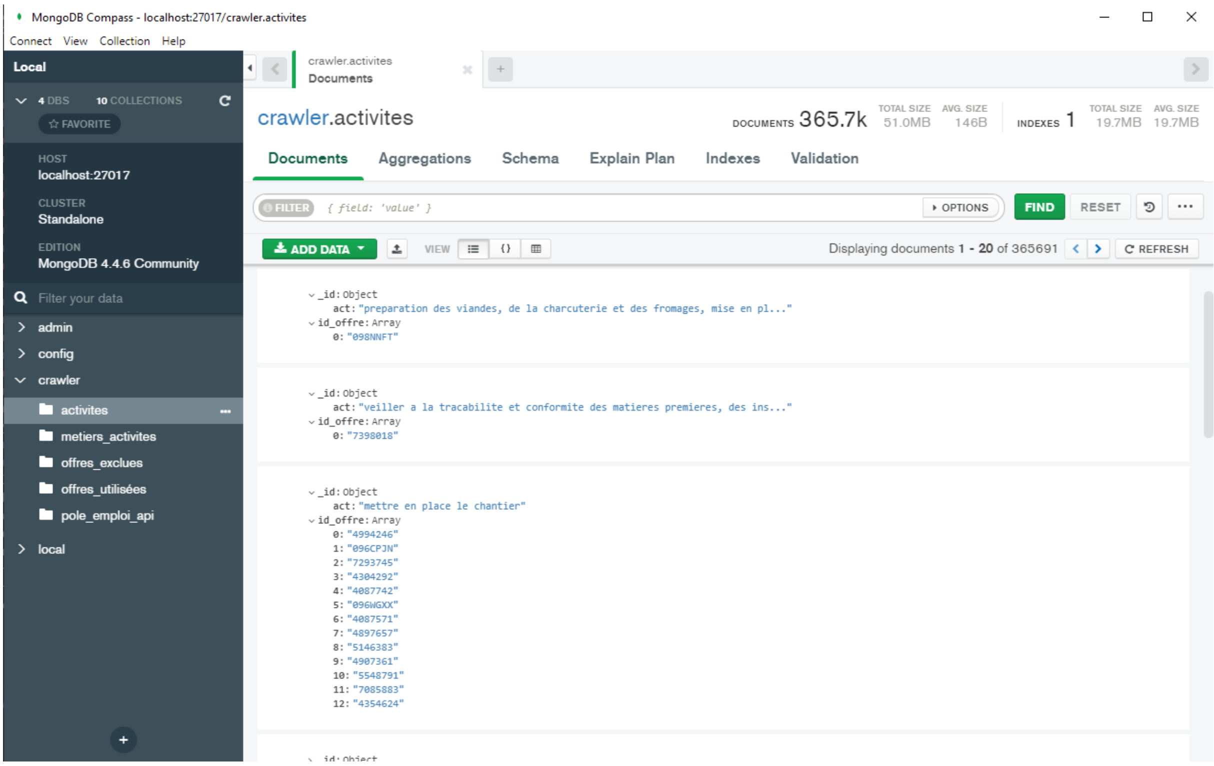


```
    [311872 rows x 1 columns]
        act_occ  compte
    0         1   43144
    1         2   26327
    2         3   16996
    3         4   12577
    4         5    9937
    5         6    8152
    6         7    6534
    7         8    5257
    8         9    4762
    9        10    4284
    10       11    3699
    11       12    3277
    12       13    3199
    13       14    2826
    14       15    2324
    15       16    2377
    16       17    2142
    17       18    2423
    18       19    2312
    19       20    1920
```



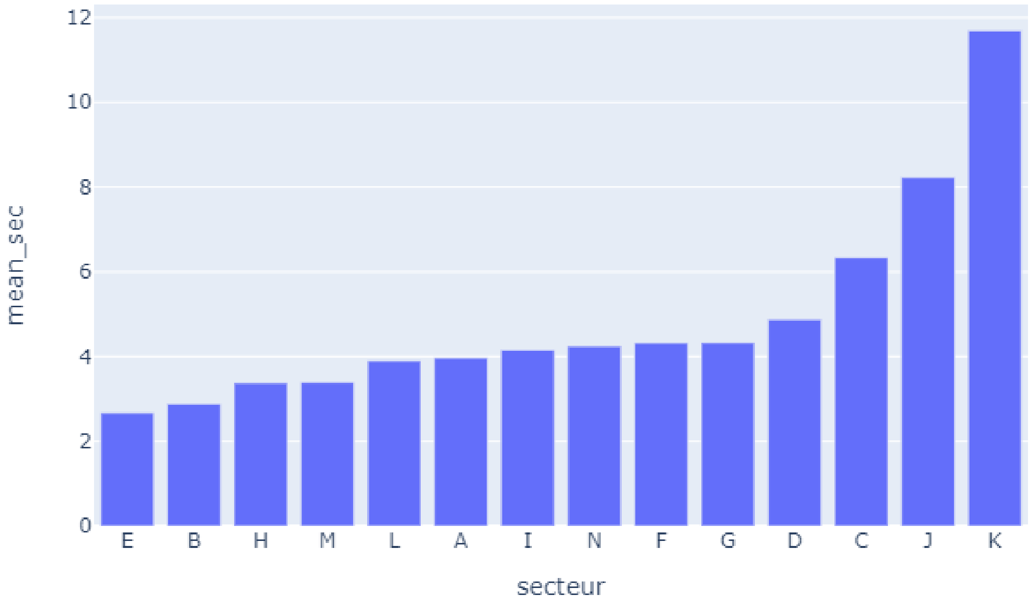
This is the histogram of the occurrence of the activities for all the offers. For example there are approximately 43000 activities which are only repeated 1 time on the 1,6 million offers.

I also created a new collection to know precisely which activities is contained in which offers



```
_id: Object
   act: "preparation des viandes, de la charcuterie et des fromages, mise en pl..."
id_offre: Array
   0: "098NNFT"

_id: Object
   act: "veiller a la tracabilite et conformite des matieres premieres, des ins..."
id_offre: Array
   0: "7398018"

_id: Object
   act: "mettre en place le chantier"
id_offre: Array
   0: "4994246"
   1: "096CPJN"
   2: "7293745"
   3: "4304292"
   4: "4087742"
   5: "096WGXX"
   6: "4087571"
   7: "4897657"
   8: "5146383"
   9: "4907361"
   10: "5548791"
   11: "7085883"
   12: "4354624"

_id: Object
```

This collection seems not very useful but in fact now I can easily find any information and all activities are well linked to an offers which was lost in the aggregation prior to that.

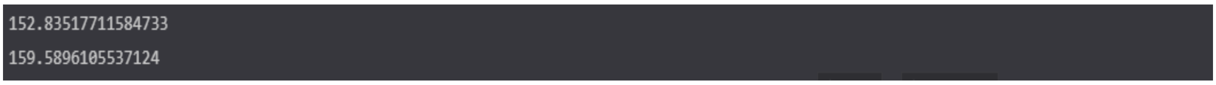


This is the mean of the occurrence of activities in each sector.

The sector K seems to be really higher in occurrence, it is because there are a few activities which have a huge amount of occurrence in this sector but it is because of the nature of the sector ”Service a la personne” that there is suc a peak.

Finally I tried to understand why some of the offers were not used to gather the activities, to try to improve this gathering if possible.

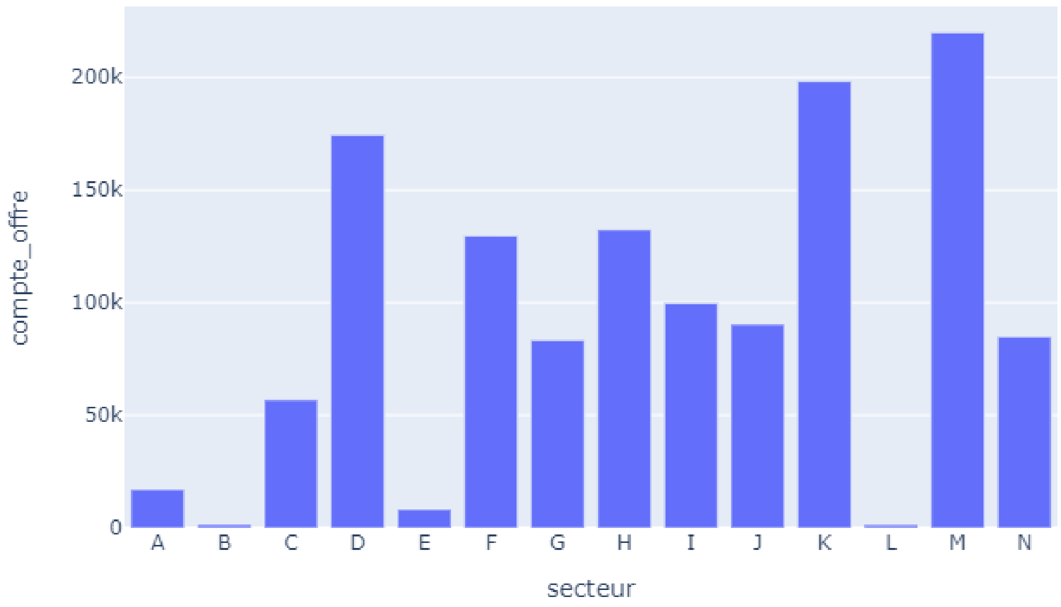
The first idea was to compare the length of the description in the offers.

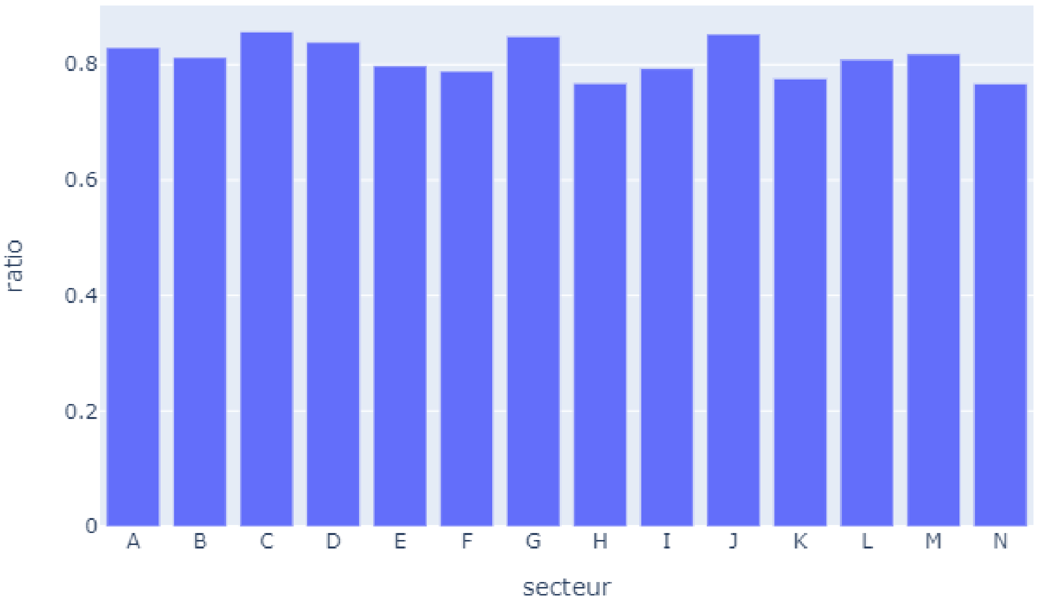


```
152.83517711584733
159.5896105537124
```

As you can see the mean of the number of words in the offers excluded and included are roughly the same so this is not the reason of the exclusion.

Then I tried to compare the different sector to see if it was correlated.





This is the different % of exclusion for the sectors

But again the ratio of exclusion is in the same spectrum of %. In the end the conclusion was that the offers excluded were either bad written or with some errors which are not useful to the project. Because it wants to help write better offers.

# VI) Conclusion

In conclusion after processing my sample of 1,6 million offers there are few points which stands out for the project.

The idea of using the activities to describe the offers seems relevant. There are some patterns and definitely the occurrence of activities shows that people are using the same terms to describe a job. Since I only worked on a 1,6 million sample there are may be some issues that will appear if the sample grows bigger. Some informations on the data are not usable due to

There are improvements and ideas that I got, for example given the geographic location people might use different words to explain a specific action. The expectation can change because of the cultural way of the area. But this are later problems since this is only the beginning of the project and such ideas will be useful to perfect the project.

# VII) Acknowledgment

I want to thanks every member of Alphaby, especially Yacine ABBOUD, who had trust in me and who take me for the internship. But everyone did take the time to explain to me exactly what were my goals during this internship. But also helping me to understand how to work in a new way in a professional environment. I learn a lot despite my work being remote I felt like I could ask anything and would be given the proper response to improve my work or to improve my understanding of the project. Thanks to Adrien ROUGERON I was able to get out of though issue that I encounter such as a massive problem to use the pymongo operation to link the data frame that I used with pandas and the mongoDB database. Thanks as well to Benjamin GRAS who take time to explain to me why some ideas that i got were not always relevant due to the application and constraint inherent to the project.